AFRL-RI-RS-TR-2007-228
Final Technical Report
October 2007

AN EXTENSIBLE ARCHITECTURE FOR MULTI-GAME FUSION

Sentar, Inc.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2007-228 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
ALEXANDER SARNACKI	JAMES W. CUSACK
Work Unit Manager	Chief, Information Systems Division
	Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* OCT 2007	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* Jun 05 – Jun 07

4. TITLE AND SUBTITLE AN EXTENSIBLE ARCHITECTURE FOR MULTI-GAME FUSION	**5a. CONTRACT NUMBER** FA8750-05-C-0210
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 62702F
6. AUTHOR(S) Dr. Leigh Flagg, Melody McClure, Dr. Rose Gamble, Dr. John Tiller	**5d. PROJECT NUMBER** 558B
	5e. TASK NUMBER SE
	5f. WORK UNIT NUMBER NT
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Sentar, Inc. 4900 University Square, Suite 8 Huntsville AL 35816-1829	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RISB 525 Brooks Rd Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2007-228

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. WPAFB # 07-0097

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The objective of the Multi-Game Fusion (MGFusion) platform is to design and prototype an extensible, reusable, and verifiable wargaming environment, which provides an advanced command and control (C2) simulation and experimentation platform. This effort was meant to provide a reasonable foundation to test C2 functions and strategies through the use of commercial-grade wargames, which strive to embody humanistic actions and responses.

15. SUBJECT TERMS
Wargame, simulation, experimentation, command and control

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Alexander Sarnacki
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UL	25	**19b. TELEPHONE NUMBER (*Include area code)***

Table of Contents

List of Figures

1 Introduction

This is constitutes the final report for the MG-Fusion project. The purpose of this document is to describe the purpose and functionality of the software platform contracted by AFRL to be used by the Air Force C2 analysts to perform advanced C2 experimentation. This document will serve as a history of the general requirements and design for the project and will outline the project's details, design issues, and components

2 MG-Fusion Design

The objective of the Multi-Game Fusion (MGFusion) platform is to design and prototype an extensible, reusable, and verifiable gaming environment, which provides an advanced command and control (C2) simulation and experimentation platform. C2 experimentation is very human factors oriented, making it difficult, time-consuming and expensive to model for integration into existing simulation platforms. This effort was meant to provide a reasonable foundation to test C2 functions and strategies through the use of commercial-grade wargames, which strive to embody humanistic actions and responses and also allow for multiple modes of play. This system would be capable of incorporating numerous and variable game engines that interact dynamically to realize any C2 scenario introduced. This promotes plug-and-play incorporation of new techniques and technologies to elucidate advanced C2 functions without system reprogramming. It would provide multiple modes of operation and interaction for C2 function experimentation. Thus, it can accommodate batch processing where a predefined scenario or campaign is introduced by a single user. In addition, a single user or team of users could directly interact with the environment to elucidate the outcome of an experimental C2 scenario. The MGFusion platform encompasses the following technologies:

Contract Templates: MGFusion employs a set of standards-based templates that serve to identify the game engines and their capabilities, define the parameters of a given simulation experiment, and return results to the platform for incorporation in the MGFusion database. These structures allow for plug-and-play interoperability of the game engines.

KnoWeb® Multi-Agent Platform: KnoWeb® is Sentar’s patented dynamic, distributed, problem-solving technology. It integrates disparate knowledge sources and employs software agents to correlate their knowledge to achieve a goal. For MGFusion, agents have been developed that interact with the contract templates, game engines, and the results database. This technology imbues the system with the characteristics of distributability, evolvability, and extensibility.

Wargames: MGFusion provides a means for integrating variable wargames by describing them to the system through the contract templates. The current testbed contains a set of 5 distinct HPS Simulations wargame engines developed by John Tiller.

Results Database: Results from wargame experiments are automatically transferred to a distributed database, where they can be further analyzed by the user’s preferred tools. The database contains all the experimental parameters such as the date, purpose, game engine, battle scenario, execution mode, game settings used as experimental variables, and game-specific outcomes.

Graphical User Interface: MGFusion provides a single, unified point of interaction between the user and the wargames. It allows the user to select a game engine and battle scenario based on the goals of the experiment, to specify settings for the experiment, and to choose mode in which to run the experiment.

2.1 MG-Fusion Requirements

MG-Fusion is an alpha prototype that allows analysts to execute both batch and interactive C2 experiments in commercial grade wargames. Performing an experiment occurs in three different stages. The first stage involves instantiating a scenario on which the experiment will ultimately will run. A user can choose from currently stored or template scenarios which they may modify using the graphical user interface. Only initial choices will be actionable, such as the type, be it ground or air campaign, of the scenario. These choices are proffered dynamically to the user and reflect capabilities that exist within currently integrated wargame engines. Furthermore, the choices made by the user in the interface affect subsequent choice availability, producing a dynamic pruning of invalid template scenarios and wargame engines that do not manifest those user inputs. Once those choices are made, appropriate template scenarios will be proffered from which a user can choose. Once all the desired category choices are modified, the user may elect to save, run or reset the scenario specification results. To launch the game, the user will select the "Run Scenario" button and its associated game will be launched according to the experiment parameters the user has defined. The next phase of system interaction occurs after the experiment has ceased its execution. When this occurs, all experiment parameters are associated with any results produced from the run and stored in the database. By the end of the program, then, the user will have chosen their scenario, used it as a basis of experimentation, run that experiment and will be provided experimentation results for use in analysis.

2.1.1 Definitions, Acronyms, and Abbreviations

Term	Definition
Scenario	Parameterization of an experiment to be executed in a wargame
Wargame	An off the shelf graphical computer game which is Battlespace and C2–centric
Project	The wargame chosen, scenario run, participants involved, tasks carried out, command hierarchy and results produced by choosing, modifying and executing a wargame.
Experiment	A session of the project in which a particular scenario may be run within a particular game multiple times to help in the analysis of a particular hypothesis
Session	A uniquely identified object associated with the results of a particular experiment executed using a particular scenario
Run	An execution of a the game.
C2	Command and control, the domain which this experimentation platform addresses. Control is *those structures and processes devised by Command to manage risk*; and Command is *the creative expression of human will necessary to accomplish a mission.*[1]
Analyst	The architect of the experiment

[1] Carol McCann & Ross Pigeau, Clarifying the Concepts of Control and of Command, Command & Control Research & Technology Symposium, U.S. Naval War College, Rhode Island, June 29 - July 1, 1999

2.1.2 MG-Fusion End User

The target audience for the MG-Fusion platform is Air Force Analysts performing C2 experiments to elucidate possible outcomes of engineered scenarios.

2.1.3 MG-Fusion Platform Perspective

- The platform is meant to be enclosed, possibly distributed, networked system and requires integration of commercial grade game engines that can be executed from a command line or console.
- The platform will incorporate wargames as they have to most pertinence to the types of experiments the analysts would like to run.
- The platform has one interface with a tabbed window: the scenario specification window, which displays the particulars of an experiment run.
- The software is designed to run on existing hardware, the hardware consisting of a server computer and multiple client computers capable of running the game instances.
- The platform requires the client computer to have a graphic capability of displaying up to 256 colors or more.
- The platform requires that the client computers run Windows and have a graphics card.

2.2 Overview of Functional Requirements

- The platform's primary focus is to allow advance C2 experimentation and archiving of the results for analysis.
- The software has 7 high-level components : **Graphical User Interface**, **User Agent**, **Meta Agent**, **Service Agent**, **Game Agent, Game(s)**, **Knowledge Store Agent**

2.2.1 General of Data Requirements

- Input Data: the software will be mouse and keyboard driven, all input data are from left mouse clicks on drop down box item, context buttons such as run, save and cancel or menu options, and text boxes into which directed information can be typed.
- Output Data: Data and information will be displayed in their respective dialogs and windows.

2.2.2 General Constraints, Assumptions, Dependencies, Guidelines

- The wargames must run on Windows 2000/XP.
- The platform should run on Linux, Windows or Unix based machine. It requires any platform that can support Java.
- The platform requires the use of a monitor, keyboard and/or a mouse as output and input devices.
- The platform requires the Wargames be installed upon client computers.
- The platform is designed for the Windows operating system but works on Unix-based and Linux platforms as well.

2.2.3 External Interface Requirements

User interface: The user interface for this platform will be graphical. It will also be context sensitive and choice driven. This is because User's are required to make many parameter choices, causing input to become complex and time consuming for the user if they are not properly guided. All input to our program will be achieved through a mouse or keyboard while all output would be via a monitor.

Hardware interaction: Our platform will be developed in Java 1.5 and XML. The games being integrated require a specific set of hardware requirements in order to function properly. Our target hardware requirements are as follows:

- 200 MHz Intel Pentium processor
- Windows 98/Me/NT/2000/XP
- 64 MB of free available system RAM
- 300 MB of available disk space
- 1024 x 768, 16-bit color display or better.
- A sound and graphics card is also recommended in order to enhance the experience. However this would not be required.

Software interaction: The platform requires Java 1.5+ be installed on all machines running MG-Fusion agents. This platform requires integration of a collection of wargames to which there is some sort of I/O interface. Ideally, they have an API such that game engine methods can be invoked when needed by MG-Fusion. It also leverages Microsoft™ Access and Excel though need not be installed to for the platform to execute.

2.3 MG-Fusion Knowledge Representation and Design

The Extensible Architecture for Multi-Game Fusion system (MG-Fusion) is a KnoWeb® based system for an extensible, reusable, and verifiable gaming environment, which provides an advanced command and control (C2) simulation and experimentation platform. The system is capable of incorporating numerous and variable game engines that interact dynamically to realize any C2 scenario introduced. To achieve this functionality, the system is be based on Sentar's proprietary KnoWeb architecture. The system utilizes the following agents: the Meta agent, service agent, user interface agent, knowledge store agent, and game agents. In addition, a common XML-based document, a contract template, will allow structured communication between wargames and the KnoWeb system. Architecture connectors, which translate game capabilities into actionable knowledge while allowing instantiation of user defined requests into consumable game scenarios, will ensure seamless bi-directional communication between game and experimentation system (see Figure 1). This section will describe the design elements of MG-Fusion including the C2 characteristics on which experimentation is based, the ontology, the role of each agent and the processing of the system incorporating the agents.

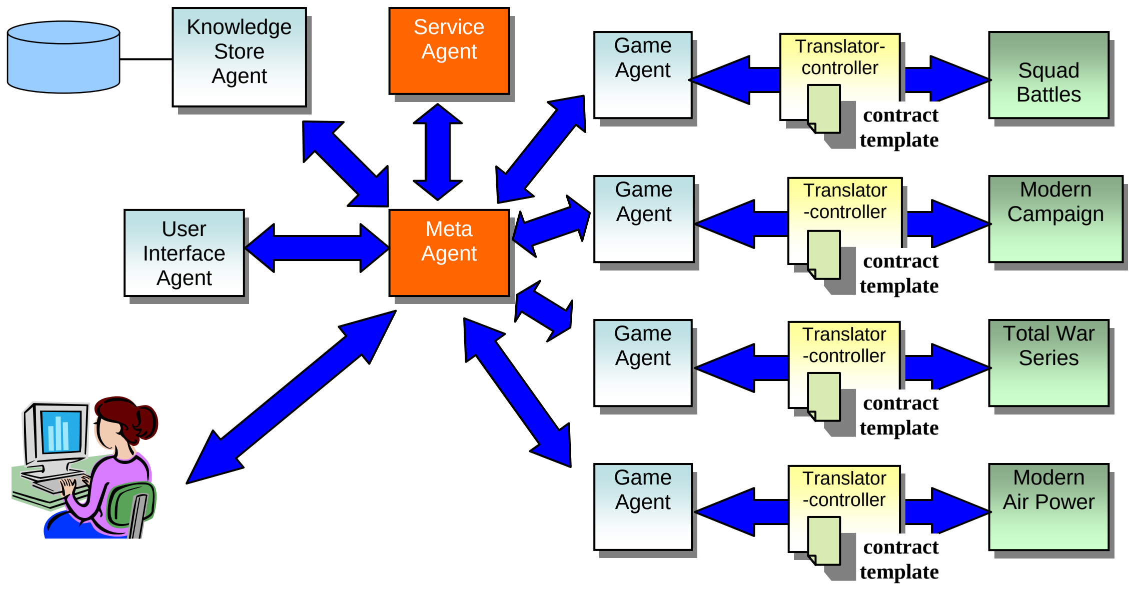

Figure 1: MG-FUSION KnoWeb® Agents

2.3.1 C2 Experimentation Characteristics

In order to utilize a critical mass of wargame engines for C2 experimentation, the experimentation characteristics of interest needed identification and normalization. Thus, both the requirements of the customer and a cross section of wargame variables were analyzed for their pertinence in C2 experimentation. The characteristics would serve two purposes: to narrow template scenarios for the user of the system such that they could identify games and template scenarios with which they might experiment. And to provide pertinent variables on which the user might make modifications to test effects. These two classes of C2 characteristics were termed Selectors and Modifiers.

These Selectors and Modifiers provide a discrete experimentation basis for any game that would be integrated into the system. In turn, they do not always directly map to all the test bed games but are represented at the game scenario level by particular variables in the scenario representation. For example, Morale is represented directly as a variable that can be modified. However, Leadership is represented in Tour of Duty scenarios as unfixed and undisrupted units.

2.3.1.1 Selectors

Air Support	Air forces available to support ground operations.
Deteriorating Operations	Operations conducted under less than optimal conditions.
Engineers	Support forces responsible for clearing mines and obstructions or construction such as bridges.
Fog-of-War	Limited information about enemy forces and friendly forces.

Force Quality	The condition of a force based on training and deterioration as a result of combat operations.
Intelligence	The ability to acquire information concerning the state of the enemy and intentions.
Preparedness	The ability to withstand an enemy attack, particularly one that would be unexpected.
Responsiveness	The ability to respond to enemy actions.
Supply	The ability to provide food, ammo, fuel, and other requirements to forces on the move and in combat.
Weather	The impact of weather on operations, including movement, combat, communication, and supply.

2.3.1.2 Modifiers

Launch Interval	The number of seconds required between successive take-offs at an airbase or aircraft carrier.
Leadership	The subjective rating given the leaders of a particular side or force.
Morale	The ability of a force to withstand adverse situations and remain effective.
Preparedness	The condition of being prepared or ill-prepared for an enemy attack.
Electronic-Warfare	The ability to detect enemy electronic communications.
Training	The subjective rating given the units of a particular side or force with respect to their skill.

2.3.2 MG-Fusion Ontology

The KnoWeb® system for MG-Fusion utilizes an ontology that, among other concepts, describes a Project, an Experiment, a Scenario and a Wargame with specified parameters. The Scenario consolidates the requests by a user to be run as a game. Some of the attributes of a scenario include the map and location, type of campaign (air or ground or naval), the forces, the tasks, and other relevant information. This ontology was defined using as a foundation the Military Scenario Definition Language. Input was also elicited from C2 analysts, the Air Force Wargame experimentation group, and by researching wargame command and control characteristics.

2.4 MG-Fusion Database Design

An important requirement of the MG-Fusion infrastructure was to make available persistent data for the user to analyze a large cross section of experiments outside the space of the experiment itself. Thus, a knowledge store was designed to record all pertinent details for an analyst seeking data possibly from experiments unknown to him. The database was programmed in Microsoft™ Access and utilizes JDBC calls to access and store information within the database as experiments are run. The tables are structured such that ontology information can be easily stored and incorporated with respect to simulations and wargames. The Microsoft™ Access database design allows for SQL queries to be written by analysts to extract useful information as needed. MG-Fusion supports export of the database to Microsoft™ Excel format so charts from data can be created as needed.

2.5 KnoWeb® System Agents

MG-Fusion utilizes the KnoWeb® system to collect attributes of a scenario, pass that information to a game engine, and start the game. When the system is initiated the Meta agent and service agent are both instantiated and started. Next the knowledge store agent, rule agents, and user-interface agent register themselves with the KnoWeb system and then start. Finally, each game agent registers itself with the KnoWeb system and is then initialized and started.

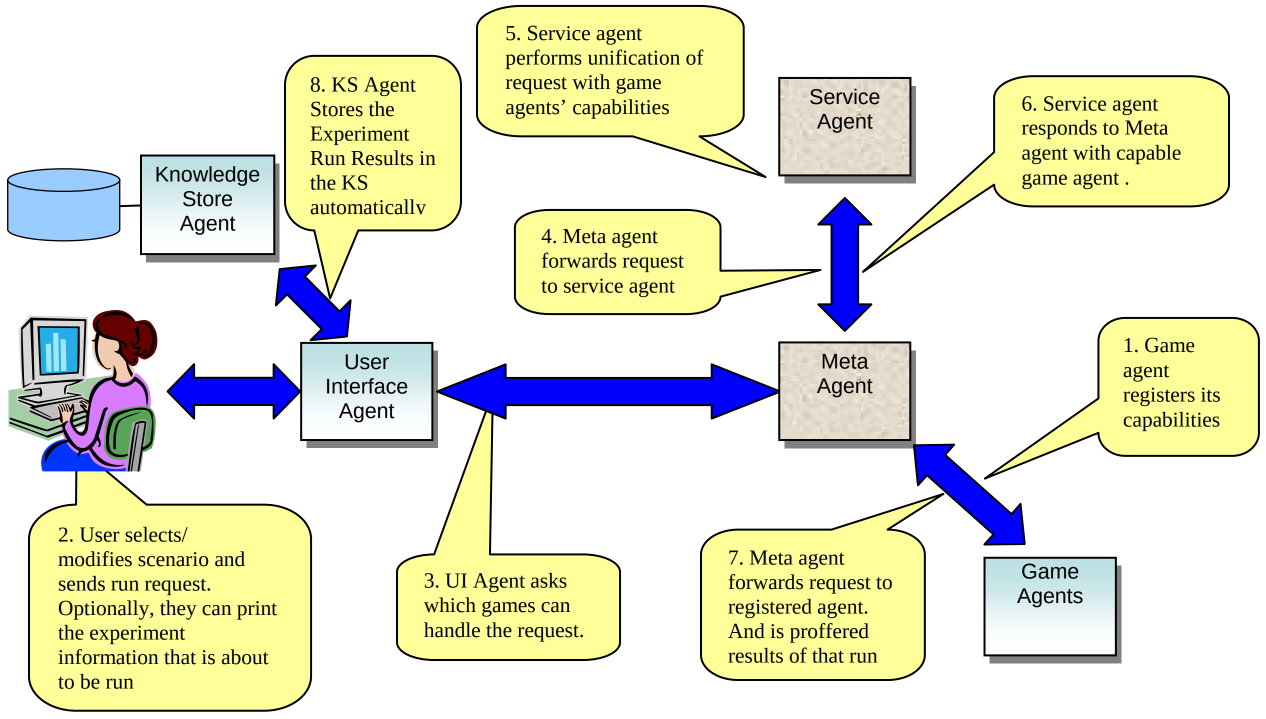
5. Service agent performs unification of request with game agents' capabilities
8. KS Agent Stores the Experiment Run Results in the KS automatically
Service Agent
6. Service agent responds to Meta agent with capable game agent .
Knowledge Store Agent
4. Meta agent forwards request to service agent
User Interface Agent
Meta Agent
1. Game agent registers its capabilities
2. User selects/ modifies scenario and sends run request. Optionally, they can print the experiment information that is about to be run
3. UI Agent asks which games can handle the request.
7. Meta agent forwards request to registered agent. And is proffered results of that run
Game Agents

Figure 2: MG-FUSION Processing

An agent registration is essentially analogous to a subscription in the publication/subscription model: it means that an agent has indicated its ability to “handle” a message with the specified content. So when a game agent registers with the KnoWeb® system, it lists the types of requests to which it has the capability to respond. Game agents in MG-Fusion register for messages that request information about supported scenarios, and for messages that issue a command to run an experiment with a given scenario. Figure 2 depicts the MG-Fusion platform overall processing which we will discuss in more detail in the subsequent sections.

2.5.1 User Interface Agent

The user interface agent is responsible for collecting information such as the scenario parameters and game operation expectations or modes from the user. In version 1.0, the user interface agent will request information about template scenarios available in the system. It will then guide the user through a process of selecting an appropriate scenario based on parameters that are important to the user. In order to define an experiment, the user will select a template scenario, modify any scenario parameters presented to them that they wish to alter, and then enter other information, such as the number, name and rank of the players, the tasks of the force, units or individual players and whether AI is to be used.

Once the user has selected a template scenario (and thereby its associated game engine), specified all other required experiment information, and issued the run command, the Meta agent will transmit the command to the registered game agent. After completion of the scenario run, the game agent will inform the Meta agent of the results. The Meta agent will then inform the knowledge store agent of the results to be stored. Finally, the Meta agent will inform the user interface agent that the scenario has been completed. The user can also use the user interface agent to view the results of the scenario or any previous scenario runs.

2.5.2 Meta Agent

The KnoWeb® Meta agent acts as a mediator for the KnoWeb system. When an agent sends a request to the KnoWeb system the Meta agent receives that request and manages that request. In the typical KnoWeb process flow, when the Meta agent receives a request, the service agent provides a list of agents that have registered for that type of request. The Meta agent then forwards the request to the registered agent(s), waits until all have sent a reply, then sends an aggregated reply back to the requestor. If there is no agent that can answer that request, the Meta agent informs the asking agent that there are no agents that can answer that request.

2.5.3 Service Agent

The Service agent is responsible for maintaining a registry of agents and their capabilities. As previously described, an agent registration is essentially analogous to a subscription in the publication/subscription model: it means that an agent has indicated its ability to “handle” a message with the specified content.

The process by which the service agent matches requests to registrations is called unification. Unification is required because the data structure in a registration is seldom exactly like the messages for which it is expressing interest, so a simple matching algorithm is not sufficient. For example, an agent might register to receive RunGame messages, where the game engine name is “War over Vietnam.” Another agent, perhaps one that stores experiments in a database, might register to receive RunGame messages, with no game engine name specified. Through the process of unification, both agents will receive a RunGame message for “War over Vietnam” but only the second agent will receive a RunGame message for “Tour of Duty.”

2.5.4 Game Agents

There will be one game agent per game engine. Each game engine maintains a set of highly-detailed template scenarios, to which the game agent has access via the contract template. When the game agent is instantiated it will register 2 capabilities with the service agent. The first is the capability to provide information about the engine and its template scenarios. The second is the capability to run the specified game engine with one or more specified scenarios.

Each game agent is connected to each game space by an external bi-directional connector. This connector:

a. Registers the game with the game agent using the contract template. In this contract template would be only the knowledge the KnoWeb® system would need to know to determine valid scenarios that game could run. Not just template scenario information and not all the template scenario information, just enough information such that the system can translate game descriptions and valid scenario information into ontology concepts to allow the GUI to direct the user as to what a valid scenario looks like to each game.
b. Receives messages from the game agent that contain modified contract templates. The connector then modifies the actual scenario and executes the scenario in the game.

The game agent interprets the prescribed values of the contract template associated with each game. Once the run request is received, the game agent uses the contract template’s XML parser to generate the actual parameters the game engine can process and return. Thus, when a game agent is requested to run a particular scenario it will pass the scenario parameters through the contract template parser to the game engine to run the game given the expected output parameters. Once the game engine has run the game, the game engine will then pass the results back to the game agent via the connector. The game agent will then pass the results back to the user interface agent via the Meta agent and on to the knowledge store agent to be archived.

2.5.5 Knowledge Store Agent

The Knowledge Store Agent is responsible for maintaining a database for experiments that have been executed and the results of such runs. The processing of the Knowledge Store agent is automatic. Upon execution of a run and its completion, all results are stored in the database for future retrieval. The knowledge store agent also exports the results of previous scenario runs, storing them in a Microsoft™ Excel spreadsheet for analysis by users. This is a menu option. Upon choice of the menu option, a macro in the Microsoft™ Access database is executed which performs the export and stores the results in the MG-Fusion folder as “Excel Spreadsheet.xls.”

2.6 *MG-Fusion Processing*

This section outlines the major user and system level use cases, System Initialization (Figure 3) and Overall System Processing (Figure 4). In the following sections, we outline the steps of the processing sequence to elucidate the high-level operations of MG-Fusion.

2.6.1 System Initialization

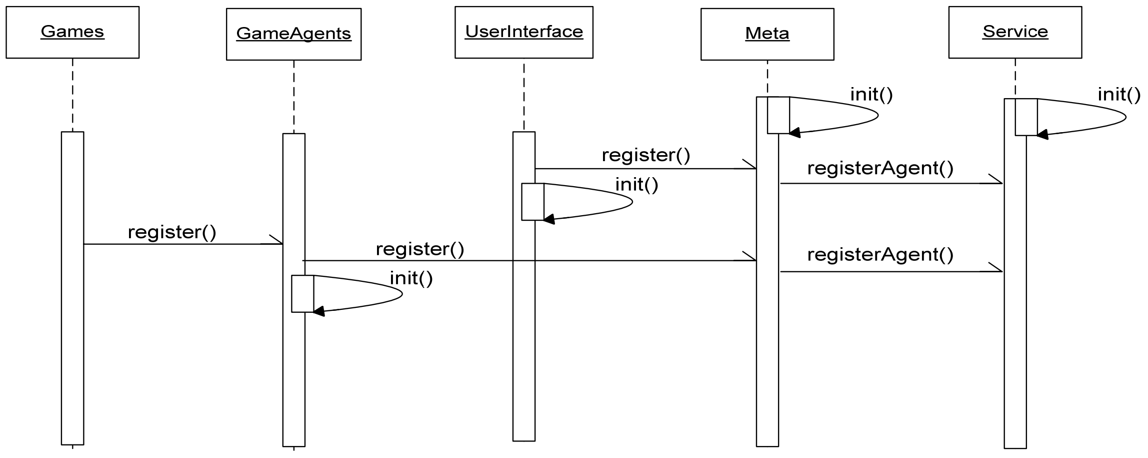

Figure 3: System Initialization Use Case Sequence Diagram

System initialization begins when:

- The Meta Agent Container and the Service Agent container initialize their respective agent configurations.
- The User Interface agent container registers its capabilities with the system (by sending a registration message to the Meta agent.)
- The Meta Agent forwards this registration on to the Service Agent who records those capabilities in a directory for future reference.
- The User Interface agent container initializes its respective agent configuration
- Each game's agent container acts as a representative for that game's capabilities and registers them with the Meta Agent.
- The Meta Agent forwards this registration on to the Service Agent who records each game's capabilities in a directory for future reference.
- The game agent containers then initialize their respective game agent configurations.

2.6.2 Overall Game Execution Processing

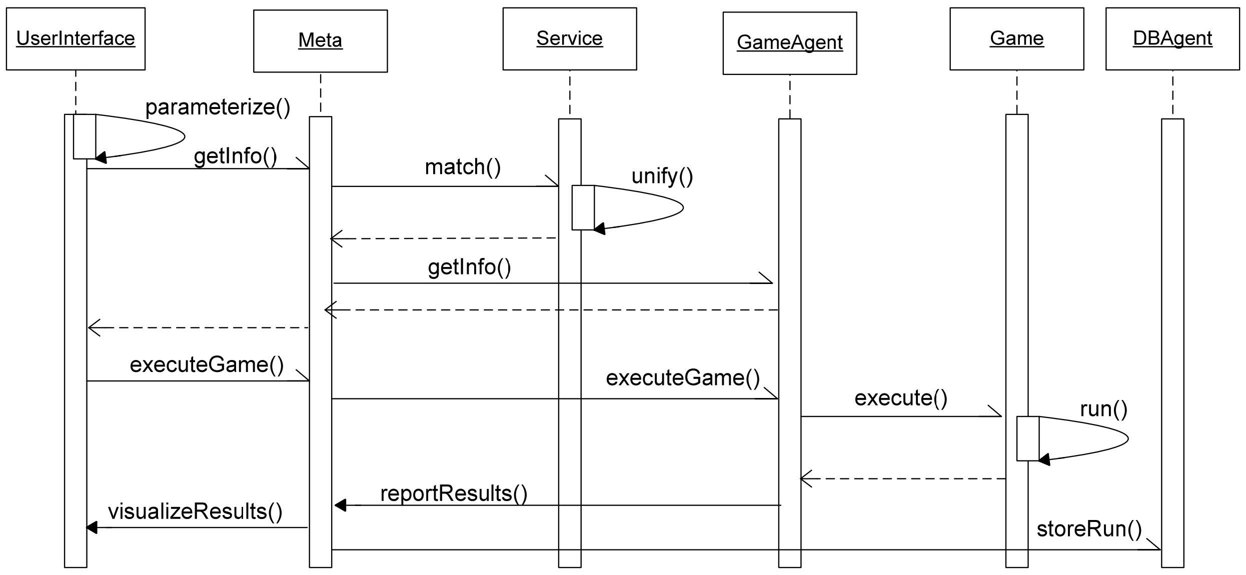

Figure 4: Overall System Processing Use Case Sequence Diagram

Overall processing of a scenario without exceptions occurs when:

- Each game's agent container acts as a representative for that game's scenario's capabilities and registers the scenario's (contract template) capabilities with the Meta Agent.
- The Meta Agent forwards this registration on to the Service Agent who records each scenario's capabilities in a directory for future reference.
- The game agent containers then initialize their respective game agent configurations.
- The User Interface agent requests information about available scenario types from all the game agents, receives a reply, and provides that information to be displayed to the user interface at the appropriate juncture in the wizard.
- The user is guided through the new experiment creation process by first using a wizard to choose appropriate parameters that are of importance for the given experiment, i.e. Service (Air/Ground), Era, Scale (Operational, Tactical, Strategic), Selectors (Fog of War, Air Support, Engineers, etc.), and Modifiers (Electronic Warfare, Launch Interval, etc.). These drill down into a very discrete subset of scenario details until a certain number of template scenarios can be chosen.
- The user chooses from that list a template scenario from one of those games to execute that experiment.
- The user then chooses a mode of experiment execution. Batch, AI versus AI or AI versus Player.
- The user now can issue the run game command, supplying the scenario metadata, such as number of runs, modifier values, etc. plus other experiment variables such as the number of participating players, their names and ranks (for recording and reporting purposes), where the instance should be run (IP addresses), etc..

- The user's specific scenario metadata including any execution specific information is sent by way of the User Interface agent to the Meta Agent. From here the Meta agent communicates an execution message to the game agent and it communicates the specific template scenario data to map the user's desired specification. The game agent reply's that a choice has been made.
- That Game agent, in turn, communicates the scenario metadata, and execution request to the game via the contract templates and instantiates the reporting facility needed.
- The game then is run by an experiment processor external to the KnoWeb architecture using the user modified scenario and execution request as the main execution parameters.

Upon Completion of scenario execution:

- The game communicates via the game loader the results of the scenario execution to its game agent.
- The game agent then forwards those results to the Meta Agent to be saved in the experiment database.
- The Meta agent forwards the results and scenario to the Knowledge Store agent to be stored.

2.6.3 Connection and Interaction

To increase the likelihood a critical mass of games would be candidates for inclusion into the MG-Fusion platform, the system required a very high degree of decoupling. In this way, games could be highly encapsulated, but their inclusion could be data driven and utilize the expressiveness of XML to translate their low level information into something interpretable by the KnoWeb® platform. This section presents the design of those data driven connection pieces for decoupling and integrating games into MG-Fusion.

2.6.3.1 Platform to Game Interface

Software integration connectors are used within MG-Fusion to facilitate transparent communication between each game engine and the MF-Fusion agents. Game engines are complicated software systems that frequently have different methods of interaction with users. Examples include input from users that required keyboard and mouse interaction, interaction with file system resources, and API calls. Connectors abstract the specific communication required by the game engine allowing MG-Fusion to setup experiments and easily execute them.

2.6.3.2 Contract Template

To incorporate unique game engine features from several different providers, MG-Fusion makes use of a unique XML encoding system referred to as a *contract template*. Each game exposes a predefined set of properties formatted to meet the expectations of the MG-Fusion agents. A contract template is created for and associated with each game; the characteristics are entered into an XML format that the MG-Fusion agents read dynamically as the user progresses through the set up of an experiment.

2.7 Game Enhancements for C2

Enhancements to the testbed games were developed to augment the command and control attributes reflected in automated game play. Specifically, two of the five testbed games were augmented: the First Blitzkrieg, and North German Plain '85. For the First Blitzkrieg a new Headquarters feature was added as well as new Isolation and Morale rules. Enhancements to North German Plain '85 include a change so that indirect fire is twice as disruptive to HQ units, and a change so that if only Detached units spot for Indirect Fire or Air Strikes, the attack is half value. Also, the Air Strike Hex Limit value was split into a separate value for each side so that C2 limitations could be represented.

2.8 Testing

A comprehensive test suite for the platform was developed as part of this effort to verify scenario execution in all modes and results database fidelity. Sample output of a test is pictured in Figure 12. Looking at Figure 5, it is important to note that all bugs were recorded, such as the **ERROR** values, and then further explained to facilitate a fix. The bugs expressed in the example below were fixed upon their reporting, and given a **VALID** status. Specifically, tests included:

- Executing all scenarios of all games in all modes on execution
- Executing multiple batch settings of all scenarios with batch-related modifiers.
- Outputting all results from the database of multiple runs to verify well-formedness of DB implementation.

Name of Game	Important Notes	Tes t #	Campaign Type	Campaign Scale	Scenario Features	Expected Results	Actual Results (1st round)	Actual Results (second round)
North German Plain	For the North German Plain game, the types should always be ground, WWII, and large (as specified by the game's Contract Template)	1	Ground	Large	N/A	All results should be of the game type North German Plain	Valid	Valid
		2	Air	Large	N/A	Results should not return the game type North German Plain	Valid	Valid
		3	Naval	Large	N/A	Results should not return the game type North German Plain	Valid (selecting WWII not an option after selecting naval)	Valid (selecting WWII not an option after selecting naval)
		4	Naval	Large	N/A	Results should not return the game type North German Plain	Valid	Valid
		5	Ground	Large	Air Support	Should return 8 results of type North German Plain	Valid	Valid
		6	Ground	Large	Deteriarating Operations	Should return 16 results of type North German Plain	Valid	Valid
		7	Ground	Large	Engineers	Should return 3 results of type North German Plain	Valid	Valid
		8	Ground	Large	Fog of War	Should return 17 results of type North German Plain	Valid	Valid

Name of Game	Important Notes	Test #	Campaign Type	Campaign Scale	Scenario Features	Expected Results	Actual Results (1st round)	Actual Results (second round)
		9	Ground	Large	Force Quality	Should return 1 result of type North German Plain	Valid	Valid
		10	Ground	Large	Intelligence	Should return 13 results of type North German Plain	Valid	Valid
Tour of Duty	For the Tour of Duty game, the types should always be Ground, Vietnam, and Large (as specified by the game's Contract Template)	1	Ground	Large	N/A	All results should be of the game type Tour of Duty	Valid	Valid
		2	Air	Large	N/A	Results should not return the game type Tour of Duty	Valid	Valid
		3	Naval	Large	N/A	Results should not return the game type Tour of Duty	Valid (selecting WWII not an option after selecting naval)	Valid (selecting WWII not an option after selecting naval)
		4	Naval	Large	N/A	Results should not return the game type Tour of Duty	Valid	Valid
		5	Ground	Large	Air Support	Should return 4 results of type Tour of Duty	Valid	Valid
		6	Ground	Large	Deteriarating Operations	Should return 4 results of type Tour of Duty	Valid	Valid
		7	Ground	Large	Fog of War	Should return 1 results of type Tour of Duty	Valid	Valid

Name of Game	Important Notes	Test #	Campaign Type	Campaign Scale	Scenario Features	Expected Results	Actual Results (1st round)	Actual Results (second round)
		8	Ground	Large	Force Quality	Should return 17 results of type Tour of Duty	Valid	Valid
		9	Ground	Large	Intelligence	Should return 1 results of type Tour of Duty	Error (User is not able to check the Intelligence option)	Error (User is not able to check the Intelligence option)
		10	Ground	Large	Preparedness	Should return 7 results of type Tour of Duty	Valid	Valid
War Over Vietnam	For the War Over Vietnam game, the types should always be Air, Vietnam, and Large (as specified by the game's Contract Template)	1	Air	Large	N/A	All results should be of the game type War Over Vietnam	Valid	Valid
		2	Ground	Large	N/A	Results should not return the game type War Over Vietnam	Valid	Valid
		3	Naval	Large	N/A	Results should not return the game type War Over Vietnam	Valid	Valid
		4	Air	Large	N/A	Results should not return the game type War Over Vietnam	Valid	Valid
		5	Air	Large	Air Support	Should return 2 results of type War Over Vietnam	Valid	Valid
		6	Air	Large	Fog of War	Should return 1 results of type War Over Vietnam	Valid	Valid

Name of Game	Important Notes	Test #	Campaign Type	Campaign Scale	Scenario Features	Expected Results	Actual Results (1st round)	Actual Results (second round)
		7	Air	Large	Force Quality	Should return 15 results of type War Over Vietnam	Valid	Valid
		8	Air	Large	Intelligence	Should return 1 results of type War Over Vietnam	Valid	Valid
		9	Air	Large	Responsiveness	Should return 14 results of type War Over Vietnam	Valid	Valid
		10	Air	Large	Supply	Should return 4 results of type War Over Vietnam	Valid	Valid
The First Blitzkrieg	For the The First Blitzkrieg game, the types should always be Ground, WWI, and Large (as specified by the game's Contract Template)	1	Ground	Large	N/A	All results should be of the game type The First Blitzkrieg	Valid	Valid
		2	Air	Large	N/A	Results should not return the game type The First Blitzkrieg	Valid	Valid
		3	Naval	Large	N/A	Results should not return the game type The First Blitzkrieg	Valid	Valid
		4	Ground	Large	N/A	Results should not return the game type The First Blitzkrieg	Valid	Valid
		5	Ground	Large	Air Support	Should return 18 results of type The First Blitzkrieg	Valid	Valid

Name of Game	Important Notes	Tes t #	Campaign Type	Campaign Scale	Scenario Features	Expected Results	Actual Results (1st round)	Actual Results (second round)
		6	Ground	Large	Deteriarating Operations	Should return 1 results of type The First Blitzkrieg	Valid	Valid
		7	Ground	Large	Engineers	Should return 2 results of type The First Blitzkrieg	Valid	Valid
		8	Ground	Large	Fog of War	Should return 5 results of type The First Blitzkrieg	Valid	Valid
		9	Ground	Large	Force Quality	Should return 5 results of type The First Blitzkrieg	Valid	Valid
		10	Ground	Large	Preparednes s	Should return 13 results of type The First Blitzkrieg	Valid	Valid
Jutland	For the Jutland game, the types should always be Naval, WWI, and Large (as specified by the game's Contract Template)	1	Naval	Large	N/A	All results should be of the game type Jutland	Valid	Valid
		2	Air	Large	N/A	Results should not return the game type Jutland	Valid	Valid
		3	Ground	Large	N/A	Results should not return the game type Jutland	Valid	Valid
		4	Naval	Large	N/A	Results should not return the game type Jutland	Valid	Valid
		5	Naval	Large	Fog of War	Should return 3 results of type Jutland	Valid	Valid

Name of Game	Important Notes	Tes t #	Campaign Type	Campaign Scale	Scenario Features	Expected Results	Actual Results (1st round)	Actual Results (second round)
		6	Naval	Large	Force Quality	Should return 2 results of type Jutland	Valid	Valid
		7	Naval	Large	Intelligence	Should return 2 results of type Jutland	Valid	Valid
		8	Naval	Large	Weather	Should return 1 results of type Jutland	Valid	Valid

Figure 5: A Sample Test Execution Output

3 Experiments

An experiment to test the efficacy of using wargames was developed for the testbed. This experiment is as follows. Testing the effectiveness of increasing Electronic Warfare (EW) where there is increased intelligence information passing and potential fog of war. MG-Fusion's scenario selection wizard was employed to select a candidate scenario. With the wizard, Fog of War and Intelligence served as the selectors. Choosing these limited us to scenarios with EW as modifiers. This is because the interface content for MG-Fusion is dynamic. So, template scenarios of current games available to the system could only modify Electronic Warfare given intelligence and fog-of-war as selectors. The game, then, that could manifest these scenarios was North German Plain '85, a future war campaign. A short campaign was chosen to minimize the running time of each execution of the batched experiment – Friend or foe along the Elbe. In order to analyze the effect of increasing EW, the experiment needed to be stepped. The decision then was to step the percentage of the NATO side's EW by 20 % starting at 0 % and increasing to the maximum at 100 %. Thus, if iterated over one repetition, this experiment would run five times.

Multiple, identical experiments were carried out with this configuration. All told, this experiment was run 5335 times. That equates to 26,675 executions of the scenario within the North German Plain '85 wargame. To analyze the data, the team endeavored to show whether NATO had a greater percentage of victories overall, and whether there was any evidence that increasing the percentage of EW for NATO increased their total points on average. This analysis was performed after running the experiment 2000 times, 4000 times, and 5335. Figures 6 and 7 provide histogram representations of the results produced at 5335 runs.

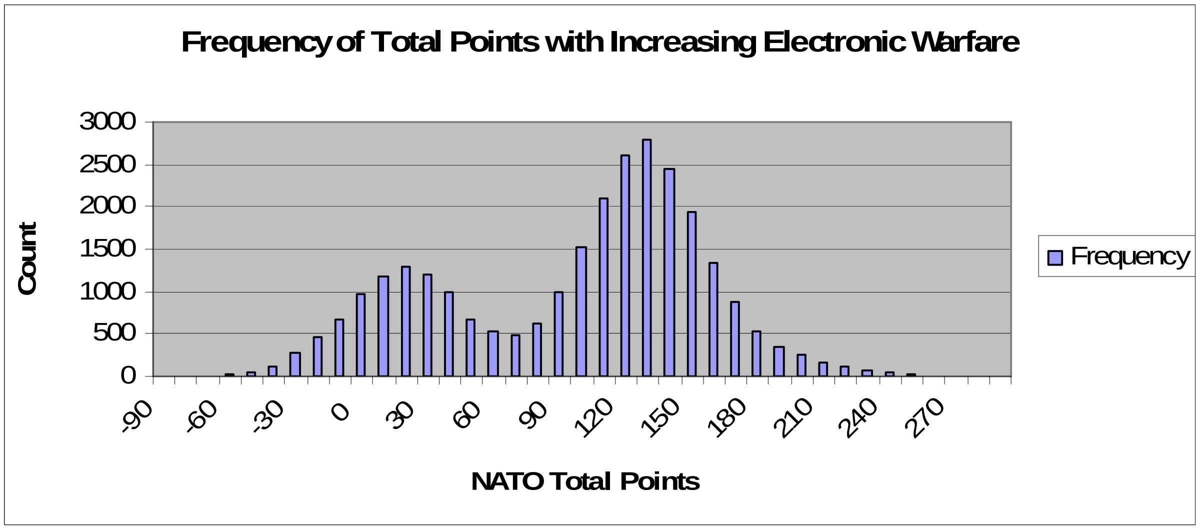

Figure 6: A Histogram Representation of the Effect of Increased EW on Total Points

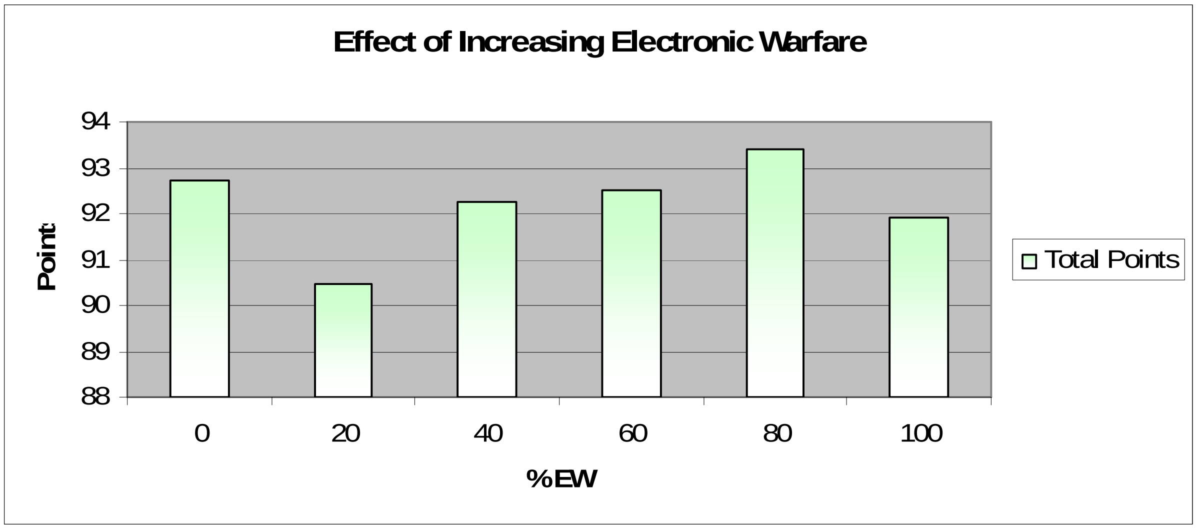

Figure 7: A Histogram Representation of the Effect of Increasing EW on Average Total Points

These results are significant as, despite the lack of a pseudo random seed, the random effects manifested in the wargames would baseline given a critical mass of runs. This is particularly evident in Figure 6 in the bi-modal results produced by charting the frequency of total point outcomes. This was also evidenced through the interim to final analysis represented in Figure 7. As more runs were conducted, the effect of EW on total points began to take on intervals of monotonic increase. At 4000 runs, a relationship between EW % and increased total points for NATO began to truly take shape starting at 40 %, where with the subsequent steps to 80 % effected an increase in total points. So far, in all cases 0 and 100 % have deviated from that pattern. This may be related to how the wargame's AI reacts to increasing EW results, firing at discovered targets more often with diminishing effects. However, it does appear to be slowly converging, indicating a slight dependence.

4 Conclusion

The result of this effort was the design and prototyping of a reflective, reusable, extensible experimentation environment, which provides advanced command and control (C2) experiment authoring, simulation execution, results archiving and analysis. This system is capable of incorporating numerous and variable game engines that interact with the user to realize experiments with C2 scenarios, running them in a computer intensive manner. Specifying a goal-driven, high-level architecture that embodies reconfigurable compositions of components ensures the ability to evolve the implemented platform. Thus, plug-and-play incorporation of new techniques and technologies to elucidate advanced command and control functions is possible without system reprogramming. The technology easily accommodates new and more complex scenarios as it is used. By designing and developing our infrastructure with the flexible, XML-based communication and employing decoupled integration enablers for external component interaction (including interaction with the games), integration completion of multiple and new game engines will be extensible in multiple dimensions as the software is extended.